Context-Only CCSLM

A Brief Technical Note for Time-Pressed Researchers

Core idea

- Consider a CCSLM system of the form

$$y = L_n(C_n, x_n) \cdots L_2(C_2, x_2) L_1(C_1, x_1) x.$$

- In **Context-Only CCSLM**, make the data input constant:

$$x = \mathbf{1}.$$

- The system therefore becomes

$$y = L_n(C_n) \cdots L_2(C_2) L_1(C_1) \mathbf{1}.$$

- The numerical input carries no information.
- **All computational variation is produced by context.**
- Context selects the linear mappings that perform the computation.

The important conceptual change

- In an ordinary data-driven system, context typically modifies how an input is processed.
- In Context-Only CCSLM, there is effectively **nothing to process**.
- Instead:

$$\boxed{C \rightarrow \text{selected computational mappings} \rightarrow y}$$

- The context is therefore not merely metadata or auxiliary information.

- It becomes the **address and organizer of computation**.

- This provides an unusually clean experimental setting for studying what happens when context is given primary responsibility for organizing a computational system.

The finite visited context set

- Suppose training encounters a finite collection of context states

$$\mathcal{C} = \{C_1, C_2, \ldots, C_m\}.$$

- Only the context states actually visited during training need to acquire useful specialized mappings.
- Training therefore establishes a collection of context-indexed computational fragments.

- Recall consists of presenting a context state and executing the corresponding chain of mappings.

- This raises an interesting question largely hidden by conventional neural-network formulations:

  What computational structure emerges when the finite set of visited context states, rather than an input distribution, is the primary organizing structure of learning?

- Context states can potentially be:
- symbolic;
- discrete sensor states;
- automaton states;
- temporal states;
- task identifiers;
- combinations of external variables;
- arbitrary externally generated state labels.

Research opportunities

Context-Only CCSLM permits experiments that are difficult to formulate naturally in conventional neural networks.

1. Context topology

- What matters is not only how many context states exist, but **which states occur together and how they transition**.
- One can study whether similar or frequently adjacent contexts develop related computational mappings.
- Context can be deliberately arranged as:
- a sequence;
- graph;
- lattice;
- finite automaton;
- cyclic state space;
- hierarchical state space.

2. Context transition dynamics

- Present contexts as trajectories rather than independent samples:

$$C_t \to C_{t+1} \to C_{t+2} \to \cdots$$

- The resulting system can be studied as a learned switched linear dynamical system.
- This creates a direct experimental connection between CCSLM, finite-state control, dynamical systems, and switched linear systems.

3. Context specialization

- Measure how different the learned mappings become for different contexts.
- Some contexts may converge toward nearly identical mappings.
- Others may develop strongly specialized mappings.
- Context states may spontaneously form **functional clusters** even when their labels are discrete and arbitrary.

4. Context capacity

- A finite number of context states provides a natural way to investigate how much computational specialization can be supported.
- One can vary:
- number of context states;
- number of mappings per layer;
- context bandwidth;
- mapping dimensionality;
- number of training visits;
- frequency distribution of contexts.
- This could reveal a form of **context capacity** distinct from conventional parameter-count or input-memory capacity.

5. Context growth

- New context states can potentially be introduced incrementally.
- This raises the possibility of allocating additional mapping capacity specifically where new contexts appear.
- Rather than increasing the entire model uniformly, computational resources could be directed toward regions of context space that require greater specialization.

6. Context-only robotics

- A robot need not always have a meaningful analog vector representing "the thing being processed."
- Sometimes the important information is simply its externally supplied state:

$$C = (\text{task}, \text{mode}, \text{phase}, \text{contact}, \text{environment}, \ldots).$$

- Context-Only CCSLM provides a natural computational architecture for such situations.
- The robot's current context can directly select the computations appropriate to that state.

A useful experimental baseline

Context-Only CCSLM also provides a particularly clean baseline for CCSLM research:

$$\boxed{\text{constant data} + \text{variable context}}$$

followed by progressively more complicated conditions:

$$\text{Context-Only} \rightarrow \text{Context-Dominated} \rightarrow \text{Mixed} \rightarrow \text{Data-Dominated}.$$

This allows the contribution of context to be experimentally separated from the contribution of the data itself.

Why this may be theoretically important

- Conventional neural networks are overwhelmingly presented as functions of their input:

$$y = f(x).$$

- Context may be incorporated, but it is generally still represented as part of the input or through mechanisms whose purpose is to process input data.
- Context-Only CCSLM instead makes it possible to investigate:

$$y = f(C)$$

where the computation is explicitly decomposed into **context-selected linear mappings**.

- The interesting object is consequently not merely the learned function f, but the **collection of computations indexed by context**.

- This creates a potentially useful experimental laboratory for questions about:

- context specialization;
- computational state;
- switched computation;
- finite-state learning;
- mapping reuse;
- context capacity;
- context trajectories;
- emergence of functional similarity between states.

Practical interpretation

Context-Only CCSLM should not be viewed as an attempt to replace ordinary data-driven computation.

It is better viewed as an unusually clean **research limit**:

> Remove information from the data pathway and ask what computational organization can emerge from context alone.

If interesting structure emerges, the experiment demonstrates something that is difficult to see when context and data are entangled.

Key phrase:
Context provides the computational address; the selected linear mappings provide the computation.